6-Implementasi Pull Message dengan menggunakan Restful Web Service pada komunikasi Wireless Sensor

by Mahar Faiq

Submission date: 31-Dec-2019 07:32PM (UTC+0700)
Submission ID: 1238861090
File name: ggunakan_Restful_Web_Service_pada_komunikasi_Wireless_Sensor.pdf (1.2M)
Word count: 4419
Character count: 26608

Register : Jurnal Ilmiah Teknologi Sistem Informasi 3(2) 65-74
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)
Tersedia *online* di www.journal.unipdu.ac.id
Unipdu
Halaman jurnal di www.journal.unipdu.ac.id/index.php/register

Implementasi *Pull Message* dengan menggunakan *Restful Web Service* pada komunikasi *Wireless Sensor*

Rakhmad Arif Hidayatullah [a], *Zamah Sari* [b], *Mahar Faiqurahman* [c]

[a,b,c] *Teknik Informatika, Universitas Muhammadiyah Malang, Malang, Indonesia*

email: [a] *kusuma.wahyu.a@gmail.com*, [b] *abdzamahsari@gmail.com*, [c] *maharf@gmail.com*

INFO ARTIKEL

Sejarah artikel:
Menerima 25 Januari 2017
Revisi 4 Mei 2017
Diterima 24 Januari 2018
Online 15 Februari 2018

Kata kunci:
Arduino
Pull Message
Restful Web Service
Wireless Sensor Network

Keywords:
Arduino
Pull Message
Restful Web Service
Wireless Sensor Network

Style APA dalam mensitasi artikel ini:
Hidayatullah, R. A., Sari, Z., & Faiqurahman, M. (2017). Implementasi Pull Message dengan menggunakan Restful Web Service pada komunikasi Wireless Sensor. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, 3(2), 65-74.

ABSTRAK

Wireless Sensor Network (WSN) merupakan jaringan dengan skalabilitas yang sangat tinggi, dan memiliki jumlah *sensor node* yang sangat banyak. Untuk efisiensi biaya, *sensor node* banyak diterapkan dengan menggunakan Arduino. Arduino merupakan papan rangkaian elektronik *open source* yang di dalamnya terdapat *chip mikrokontroler*. Dengan keterbatasan *resource* yang dimiliki oleh Arduino, efisiensi komputasi yang ada di dalam *sensor node* harus diperhatikan. Salah satunya berkaitan dengan proses komunikasi dan pengiriman data dari *sensor node* ke *sink node* (*gateway*). *Restful web service* merupakan salah satu protokol komunikasi yang memanfaatkan HTTP. Protokol ini dikenal memiliki efisiensi yang cukup tinggi, di samping karena interoperabilitasnya untuk digunakan pada berbagai *platform*. Dalam makalah ini akan diuraikan hasil dari implementasi model *pull message* dengan menggunakan *restful web service*, pada komunikasi antara *sink node* (*gateway*) dengan *sensor node*, di dalam infrastruktur WSN. Dalam penelitian yang dilakukan, digunakan mikrokontroler Arduino sebagai *sensor node*, dan Raspberry Pi sebagai *sink node*. Selain itu juga diimplementasikan mekanisme *thread* untuk menangani *multi-process* yang berjalan di dalam *sensor node*. Hasil dari pengujian menunjukan bahwa interval *sensing*, ukuran data, dan jumlah *sink node* yang melakukan *request*, tidak begitu berpengaruh terhadap ketersedian *free memory heap* pada *sensor node*. Sedangkan ukuran dari data hasil *sensing* yang dikirim mempunyai pengaruh terhadap *request-response time*.

ABSTRACT

Wireless Sensor Network (WSN) infrastructure has very high scalability, which is composed of a large number of the sensor node. For cost efficiency, sensor node broadly deployed using Arduino. Arduino is the open-source electronic circuit board, which has embedded microcontroller chipset. Due to limited resource, the efficiency of computation inside Arduino must be considered during the development of sensor node. One of them is related to communication and data delivery process between the sensor node and sink node. Restful Web Service is one of communication protocol framework which uses HTTP protocol and claimed to be the most efficient Web Service. As well as the other web services, Restful also support high interoperability of communication. In this paper we describe the implementation of pull message mechanism on WSN communication between the sensor node and sink node, using Restful Web Service. We implemented sensor node using Arduino board, and sink node using Raspberry Pi. We also used thread mechanism to handle multi-process that run in the sensor node. The results of the study show that the interval time in sensing, data size, and the number of sink node which makes requests, didn't give too much influence in the availability of free memory heap in the sensor node. While the size of sensor data that was sent to sink node have an influence on a request-response time.

ISSN 2502-3357 (Online)
R. A. Hidayatullah dkk./Register 3(2) 65-74 ISSN 2503-0477 (Print)

1. Pendahuluan

Wireless Sensor Network (WSN) merupakan suatu infrastruktur jaringan nirkabel yang terdiri dari beberapa *node* yang dapat bertindak sebagai *sensor* maupun sebagai *sink* (*gateway*). *Sensor node* bertanggung jawab di dalam pengambilan data dari lingkungan, sedangkan *sink node* bertindak sebagai pengumpul data dari *sensor node* untuk dikirimkan ke *server database* atau pengolah data. Teknologi WSN dapat digunakan untuk memonitor beberapa hal seperti temperatur, kelembapan, jarak benda, cahaya dan lain sebagainya dan digunakan untuk tujuan tertentu. Beberapa penerapan *WSN* yang digunakan untuk sistem *monitoring*, diantaranya adalah penggunaan WSN untuk *monitoring greenhouse* (Sawidin, Melo, & Marsela, 2015; Chaudhary, Nayse, & Waghmare, 2011), *monitoring* nutrisi air pada aquaponik (Maarif, 2016), dan *monitoring* kondisi lingkungan (Ferdoush & Li, 2014).

Dalam implementasinya saat ini, pengembangan WSN banyak yang memanfaatkan *board* mikrokontroler yang bersifat *Open Source*, diantaranya adalah Arduino. Arduino ini memiliki beberapa keunggulan, diantaranya adalah harga perangkat yang tidak terlalu mahal, dan dapat diintegrasikan dengan beberapa modul *hardware* yang lain, termasuk modul *sensor* (Georgitzikis, Akribopoulos, & Chatzigiannakis, 2012). Arduino ini juga memiliki kemampuan yang cukup baik untuk menjalankan proses *sensing* dan pengiriman data hasil *sensing*. Beberapa penelitian sebelumnya yang memanfaatkan Arduino sebagai *sensor node* diantaranya adalah pada sistem *monitoring* kondisi lingkungan dengan memanfaatkan WSN (Ferdoush & Li, 2014). Selain itu terdapat penelitian lain yang juga memanfaatkan Arduino yaitu pada sistem WSN untuk *monitoring* suhu dan kelembaban (Barroca, et al., 2013).

Karena *resource* yang dimiliki oleh *sensor node* yang berbasis Arduino sangat terbatas, maka komputasi yang terdapat di dalamnya harus efisien. Termasuk komputasi yang berkaitan dengan proses komunikasi antar *node* dan antara *sensor node* dengan *sink node*. Selain itu permasalahan interoperabilitas komunikasi juga menjadi tantangan dalam pengembangan WSN dengan *resource* yang terbatas. Beberapa penelitian telah dilakukan sebelumnya terkait dengan mekanisme komunikasi antar *node* dalam WSN. Othman, Glitho, & Khendek (2007) dalam penelitiannya telah mengimplementasikan SOAP *Web Service framework* untuk mekanisme komunikasi antar *node*. Hasil dari penelitian tersebut menunjukan bahwa penggunaan *web service* dapat meningkatkan interoperabilitas komunikasi, namun kurang efisien dalam penggunaan *resource* yang ada pada tiap *node*. Sementara pada penelitian yang lain dikembangkan suatu model *Multi-level SOA* yang dapat mengefisiensikan komunikasi berdasarkan pada sumber daya *hardware* yang dimiliki oleh *sensor node* (Leguay, Lopez-Ramos, Jean-Marie, & Conan, 2008). Sedangkan Kabisch dalam makalahnya memaparkan efisiensi dari penggunaan format data XML untuk pertukaran data dalam WSN (Käbisch, Peintner, Heuer, & Kosch, 2010).

Restful Web Service merupakan salah satu teknologi *web service* yang memanfaatkan protokol HTTP untuk pertukaran datanya. *Restful Web Service* banyak digunakan saat ini, karena selain mendukung interoperabilitas seperti *Web Service* lainnya, juga mempunyai efisiensi komunikasi yang tinggi. Pada penelitian yang telah dilakukan (Hamad, Saad, & Abed, 2010), disebutkan bahwa *Restful Web Service* ini sangat baik diimplementasikan pada *hardware* yang memiliki *resource* terbatas seperti pada perangkat *mobile*. Dengan sifat dari *Restful* yang *stateless* seperti halnya HTTP *request-response*, serta memiliki ukuran data yang cukup kecil, *Web Service* ini dapat menghemat penggunaan *resource* komputasi untuk proses pengiriman data.

Pola dari pengiriman data juga berpengaruh terhadap penggunaan *resource* pada *sensor node*. Secara garis besar, pola pengiriman data dibagi menjadi dua, yaitu *push message* dan *pull message*. *Push message* akan dikirimkan secara langsung oleh penyedia data ke *client* yang membutuhkan data tanpa didahului oleh proses permintaan data dari *client* tersebut. Sedangkan pada *pull message*, data akan diberikan oleh penyedia data hanya ketika ada permintaan data dari *client*. Dari penelitian yang telah dilakukan sebelumnya (Burgstahler, Lampe, Richerzhagen, & Steinmetz, 2013), didapatkan bahwa pada kasus tertentu model *push-message* memiliki efisiensi yang lebih rendah dalam penggunaan sumber daya, dibandingkan dengan *pull-message*. Hal ini disebabkan karena pada *push message*, koneksi antara sumber data dengan *client* yang membutuhkan data dibuka secara terus menerus, meskipun *client* tidak sedang memerlukan data. Sedangkan pada *pull message*, koneksi dilakukan hanyak ketika *client* membutuhkan data dari sumber data.

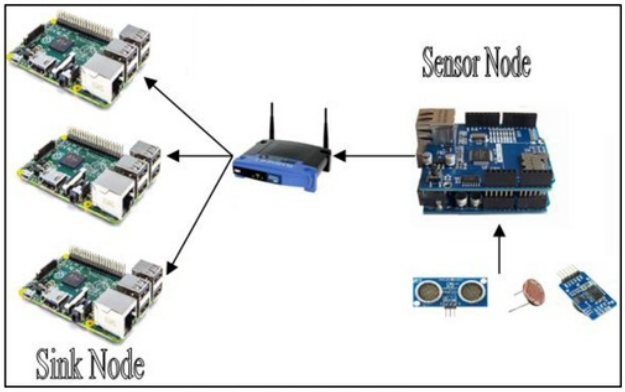
ISSN 2502-3357 (Online)
R. A. Hidayatullah dkk./Register 3(2) 65-77 ISSN 2503-0477 (Print)

Dalam makalah ini dibahas mekanisme *pull message* yang diimplementasikan dengan menggunakan *Restful Web Service* sebagai protokol komunikasi antara *sensor node* dan *sink node*. Dalam penelitian yang dilakukan, digunakan mikrokontroler Arduino untuk implementasi *sensor node* dan Raspberry Pi untuk implemetasi *sink node*. Proses *pull message* diinisiasi oleh beberapa *sink node* yang melakukan permintaan data hasil *sensing* ke *sensor node*. Sehingga setiap *sensor node* harus dapat menangani permintaan oleh beberapa *sink node* dalam satu waktu.

2. Metode Penelitian

2.1. Arsitektur system

Arsitektur sistem yang dibangun dapat dilihat pada Gambar 1, terdiri dari beberapa komponen utama, yaitu *sensor node* yang diimplementasikan menggunakan Arduino dan *sink node* yang diimplementasikan menggunakan Rapsberry Pi. *Sensor node* bertanggung jawab untuk melakukan proses *sensing* (mengambil data dari lingkungan dengan menggunakan sensor), dan mengirimkan data hasil *sensing* ketika *sink node* melakukan proses *pull-message*. Sedangkan *sink node* bertanggung jawab untuk mengumpulkan data hasil *sensing* dari *sensor node* melalui mekanisme *pull message*. Proses *pull message* ini dilakukan ketika *sink node* membutuhkan data hasil dari proses *sensing*. Data ini selanjutnya akan dikirimkan oleh *sink node* ke server untuk diolah. Karena penelitian yang dilakukan berfokus pada komunikasi antara *sensor node* dengan *sink node* menggunakan mekanisme *pull-message*, maka segala hal yang berkaitan dengan proses pengolahan data yang ada di server, serta komunikasi antara *sink node* dengan server diasumsikan tidak dibahas dalam sistem.



Gambar 1. Arsitektur sistem

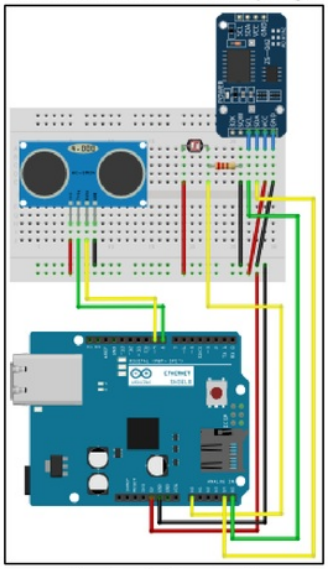
Mekanisme komunikasi antara *sensor node* dengan *sink node* dilakukan melalui medium nirkabel dengan perantara *access point*. Satu buah *sensor node* dapat terkoneksi dengan beberapa buah *sink node* yang akan melakukan proses *pull-message* data hasil *sensing*. Hal ini berdasarkan asumsi bahwa data hasil *sensing* yang dimiliki oleh suatu *sensor node*, dibutuhkan oleh beberapa *sink node*. Protokol Restful Web Service digunakan selama proses komunikasi berlangsung antara *sensor node* dengan *sink node*. Di dalam *sensor node* terdapat HTTP server sederhana yang dapat menerima *request* data hasil *sensing* dari *sink node*, dengan menggunakan mekanisme *pull message*. Respon yang berisi data hasil *sensing* diberikan oleh *sensor node* dalam format *Restful*. Karena interval waktu proses *pull message* yang dilakukan oleh beberapa sink node diasumsikan tidak terjadi secara tetap, maka sensor node perlu menyimpan data hasil *sensing* selama beberapa waktu di dalam memori internal, sampai data tersebut diambil oleh semua *sink node*.

2.2. Perancangan *hardware* untuk *sensor node* dan *sink node*

Komponen *hardware* dalam sistem yang dibangun terdiri dari Arduino, Raspberry Pi, dan *Wireless Access Point* (WAP). Mikrokontroller Arduino digunakan sebagai *sensor node* yang melakukan proses *sensing* dengan menggunakan *sensor*. Raspberry Pi digunakan sebagai *sink node* yang mengumpulkan

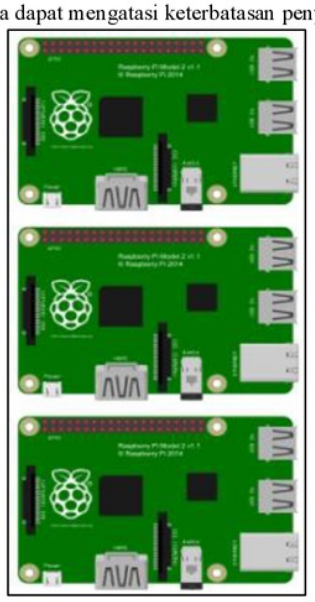
ISSN 2502-3357 (Online)
R. A. Hidayatullah dkk./Register 3(2) 65-74 ISSN 2503-0477 (Print)

data dari *sensor node* melalui mekanisme *pull-message*. Sedangkan WAP digunakan sebagai perantara komunikasi antara *sensor node* dengan *sink node*. Dalam penelitian ini digunakan WAP agar satu buah *sensor node* dapat menerima koneksi lebih dari satu *sink node* yang membutuhkan data hasil *sensing*.



Gambar 2. *Sensor node*

Skema *hardware* untuk *sensor node* dapat dilihat pada Gambar 2. Arduino yang merupakan komponen utama dari *sensor node* terhubung dengan modul *sensor*. Karena fokus penelitian terdapat pada mekanisme komunikasi antara *sensor node* dengan *sink node*, maka dalam penelitian yang dilakukan diasumsikan digunakan *sensor* cahaya *Light Dependent Resistor* (LDR), dan *sensor* jarak (*ultrasonic*). Sensor yang digunakan tersebut terhubung dengan *port* GPIO dari Arduino, dan beberapa *port* lain seperti *power* dan GND. Selain itu terdapat modul *real time clock* (RTC) yang terhubung dengan pin SLC dan SDA atau pin A5 dan A4 pada Arduino. Fungsi dari modul RTC adalah untuk menangani pewaktuan pada sistem yang dapat menghasilkan *current time* yang digunakan dalam pencatatan data hasil *sensing*. Selain itu, di Arduino juga terdapat modul *ethernet shield* yang digunakan untuk koneksi dengan WAP. SD *card* yang terdapat pada *Ethernet Shield* dapat dimanfaatkan untuk menyimpan data hasil *sensing*, sehingga dapat mengatasi keterbatasan penyimpanan yang terdapat pada Arduino.



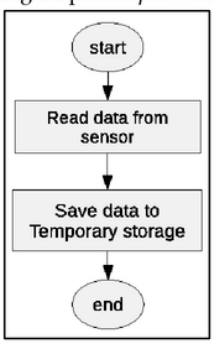
Gambar 3. *Sink node (gateway)*

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

Skema *hardware* dari *sink node* dapat dilihat pada Gambar 3. *Sink node* yang juga berperan sebagai *gateway* diimplementansikan dengan menggunakan Raspberry Pi. Dalam penelitian ini dipilih Raspberry Pi untuk *sink node*, karena proses yang berjalan di dalam *sink node* tidak terlalu kompleks, dan membutuhkan *storage* yang besar. *Sink node* ini terkoneksi dengan *sensor node* dengan menggunakan jaringan nirkabel, melalui modul Wi-Fi yang terdapat pada Raspberry Pi. Sedangkan LAN *port* dari Raspberry Pi, terkoneksi dengan jaringan kabel ke server pengolah data. Namun dalam penelitian ini diasumsikan bahwa koneksi dari Rapsberry Pi ke server pengolah data, dan proses yang ada di pengolah data tidak dibahas. Di dalam Raspberry Pi, segala data sensor hasil permintaan ke sensor *node* akan disimpan ke dalam SD *card*. Data ini kemudian akan dikirimkan ke server untuk diolah.

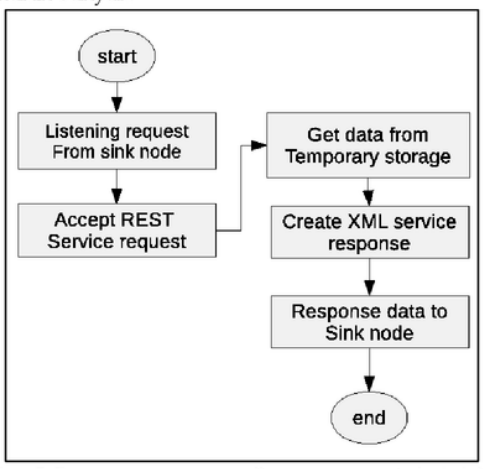
2.3. Proses *sensing* dan penanganan *Pull Message*

Proses yang terdapat pada *sensor node* di Arduino dibagi menjadi dua bagian utama, yaitu proses yang menangani *sensing* untuk mengambil data dari lingkungan, dan proses untuk menerima permintaan data melalui mekanisme *pull message* dari *sink node*. Kedua proses tersebut dikerjakan secara paralel, di mana *sensor node* dapat menerima dan melayani proses *pull message* dari *sink node* pada waktu *sensor node* sedang melakukan proses *sensing*. Demikian juga untuk penanganan proses *pull message*, dalam sekali waktu *sensor node* juga dapat menangani proses *pull message* dari beberapa *sink node* sekaligus.



Gambar 4. Proses *sensing* pada *sensor node*

Mekanisme *sensing* yang terdapat pada *sensor node* dapat dilihat pada Gambar 4. Dalam proses *sensing* dilakukan pembacaan *sensor* LDR dan *ultrasonic* secara periodik. Proses pembacaan *sensor* LDR dan *ultrasonic* dilakukan secara berurutan. Setelah proses pembacaan data oleh *sensor* dilakukan, selanjutnya data disimpan dalam *temporary storage* yang berupa SD *card* yang terdapat di dalam modul *Ethernet Shield* sebagai unit penyimpanan lokal. Dengan memanfaatkan SD *card* ini data hasil *sensing* yang dapat ditampung semakin banyak.



Gambar 5. Proses penanganan *pull-message request* dari *sink node*

https://doi.org/10.26594/register.vi.idpublikasi

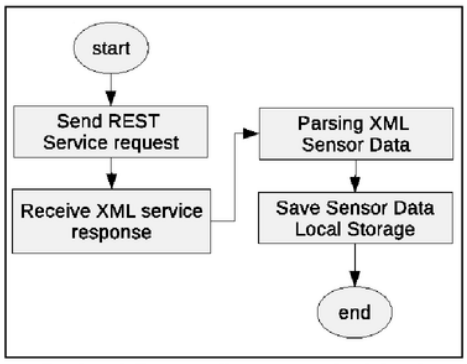
ISSN 2502-3357 (Online)
R. A. Hidayatullah dkk./Register 3(2) 65-74 ISSN 2503-0477 (Print)

Gambar 5 menunjukan proses penanganan *request* data dari *sink node* melalui mekanisme *pullmessage*. *Request* data diterima oleh *sensor node* dalam bentuk *REST service request* melalui protokol HTTP. Sehingga untuk menangani *request* tersebut, di dalam *sensor node* diimplementasikan HTTP *server* yang terus berjalan melakukan proses *listening* permintaan data dari *sink node*. Ketika menerima *request*, *HTTP server* ini akan mengembalikan *response* kepada *sink node* berupa data hasil *sensing* dalam format *XML service response*. Data hasil *sensing* yang dikirimkan tersebut ini tidak langsung didapatkan dari modul *sensor* secara *real time*, akan tetapi diambil dari *temporary storage* yang terdapat pada *SD card* di dalam modul *Ethernet Shield*. Hal ini dilakukan karena semua data hasil *sensing* ditampung terlebih dahulu di dalam *temporary storage* tersebut.

Di dalam *sensor node* juga diimplementasikan *thread*. *Thread* ini digunakan untuk menangani eksekusi beberapa proses yang ada dalam *sensor node* agar dapat berjalan secara *paralel*. Proses yang ditangani oleh *thread* diantaranya adalah proses *sensing* dan proses-proses yang terdapat pada *web server* seperti proses *listening request* dan proses *response* data hasil *sensing*. Dikarenakan secara *hardware*, unit pemrosesan data yang terdapat pada Arduino tidak mendukung *thread*, maka pemrosesan *thread* dilakukan dengan konsep *time slicing*, di mana beberapa proses akan dieksekusi secara bergantian sesuai dengan interval waktu tertentu.

2.4. Proses *Pull Message*

Proses *pull message* merupakan proses permintaan data sensor oleh *sink node* ke *sensor node*, dengan cara mengirimkan pesan *request* ke *sensor node*, dan setelah itu *sink node* akan menerima *response* berupa data sensor dari *sensor node*. Gambar 6 menunjukan proses *pull message* yang terdapat pada *sink node*. Seperti yang dijelaskan pada Gambar 6, ketika *sink node* menginginkan data hasil *sensing*, maka *sink node* akan mengirimkan *REST service request* kepada pada *sensor node*, dengan menyertakan informasi *log* waktu data hasil *sensing* terakhir kali didapatkan. Informasi waktu ini nantinya digunakan oleh *sensor node* untuk menentukan data manakah yang akan dikirim ke *sink node*. Dengan mekanisme seperti ini, *sensor node* tidak perlu mengirimkan semua data hasil *sensing* yang disimpan di dalam SD *card*, akan tetapi cukup mengirimkan data terbaru yang belum didapatkan oleh *sink node*. Setelah mendapatkan *response* dari *sensor node* dalam format XML, *sink node* akan melakukan *parsing* pada data *response* tersebut dan memasukannya ke dalam *database* yang terdapat pada *sink node*. Data ini selanjutnya akan dikirimkan ke *server* untuk diolah.



Gambar 6. Proses *pull message* pada *sink node*

3. Hasil Penelitian dan Pembahasan

Setelah mekanisme *pull message* dengan menggunakan *Restful Web Service* berhasil diimplementasikan di dalam proses komunikasi antara *sensor node* yang berbasis Arduino, dan *sink node* yang berbasis Raspberry Pi, maka dilakukan pengujian terhadap sistem *wireless sensor* tersebut. Pengujian dilakukan untuk mengetahui performa dari *sensor node* yang dipengaruhi oleh beberapa parameter pengujian. Karena keterbatasan fitur pada *monitoring* sistem yang terdapat di Arduino, maka ukuran performa dari Arduino dilihat dari banyak sedikitnya *memory heap* yang tersisa di dalam Arduino selama

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

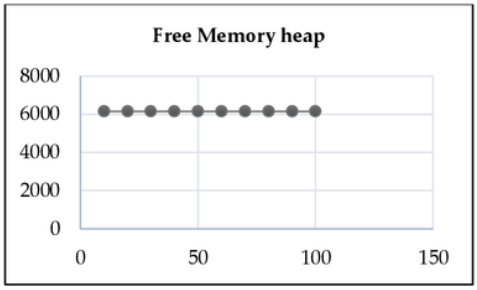
menjalankan proses komputasi. Ada beberapa skenario pengujian yang dilakukan, yaitu: pengujian untuk mengetahui pengaruh ukuran data yang dikirim pada mekanisme *pull message* terhadap *memory heap* yang tersisa, pengujian untuk mengetahui pengaruh jumlah *sink node* yang melakukan *request* terhadap *memory heap* yang tersisa, dan pengujian pengaruh besar data terhadap waktu *request-response* pada mekanisme *pull message*.

3.1. Pengujian pengaruh ukuran data pada *pull message* terhadap *memory heap*

Pengujian ini dilakukan pada *sensor node*. Ukuran data yang dikirim oleh *sensor node* ke *sink node* melalui *pull message* tergantung dari seberapa sering *sensor node* melakukan proses *sensing* (interval waktu *sensing*). Tujuan pengujian ini untuk mengetahui apakah ukuran data sensor yang dikirim sebagai *response* ke *sink node* di dalam mekanisme *pull message*, akan berpengaruh terhadap *memory heap* yang tersisa di *sensor node*. Pencatatan *memory heap* yang tersisa dilakukan oleh *thread* tersendiri yang berjalan secara paralel dengan proses pengiriman *response* di dalam mekanisme *pull message*. Skenario pengujian ini dengan dilakukan beberapa percobaan proses *pull* data hasil *sensing* oleh *sink node* dengan variasi ukuran data yg berbeda, dimulai dari 10 KB, 20 KB, 30 KB, 40 KB, 50 KB, 60 KB, 70 KB, 80 KB, 90 KB, dan 100 KB. Hasil pengujian dapat dilihat pada Tabel 1 dan Gambar 7.

Tabel 1. Hasil pengujian pengaruh ukuran data

No.	Ukuran Data (Kilobyte)	*Free Memory Heap*
1	10	6148
2	20	6148
3	30	6148
4	40	6148
5	50	6148
6	60	6148
7	70	6148
8	80	6148
9	90	6148
10	100	6148



Gambar 7. Grafik hasil pengujian pengaruh ukuran data

Hasil yang didapat dari pengujian ini dapat dilihat pada Gambar 7, dapat dilihat bahwa besarnya data yang dikirim tidak berpengaruh terhadap *memory heap* yang tersisa di dalam *sensor node*. Dari hasil analisis yang telah dilakukan, hal ini disebabkan karena tidak adanya mekanisme sinkronisasi dan penjadwalan *thread* menyebabkan proses pencatatan *memory heap* yang tersisa tidak dapat dilakukan pada waktu yang benar-benar bersamaan dengan proses pengiriman data *response* berlangsung. Sehingga ada kemungkinan nilai yang tercatat di hasil pengujian merupakan nilai *memory heap* yang tersisa pada saat *sensor node* (mikrokontroler Arduino) tidak sedang mengeksekusi proses pengiriman data *response*.

3.2. Pengaruh jumlah *sink node* yang melakukan *pull message* terhadap *memory heap*

Pengujian ini dilakukan pada *sensor node*. Tujuan pengujian ini untuk mengetahui pengaruh jumlah *sink node* (Raspberry Pi) yang melakukan *request* data dengan *pull message* terhadap *memory heap* yang tersisa

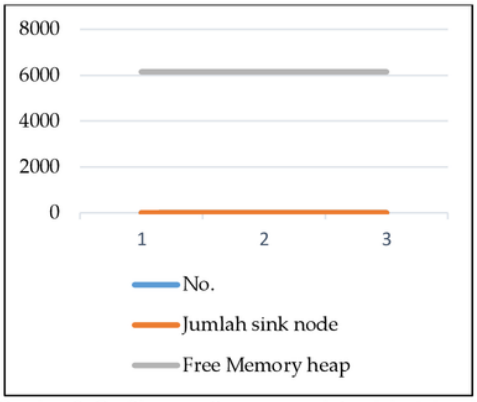
https://doi.org/10.26594/register.v1i.idpublikasi

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

pada *sensor node* (Arduino). Seperti pengujian sebelumnya, pencatatan *memory heap* yang tersisa dilakukan dengan menggunakan *thread* yang terpisah, akan tetapi berjalan secara paralel dengan *thread* yang menangani proses penerimaan *request*. Skenario pengujian dilakukan dengan cara melakukan variasi pada jumlah *sink node* yang melakukan *request* ke *sensor node*. Hasil pengujian ini dapat dilihat pada Tabel 2 dan Gambar 8.

Tabel 2. Hasil pengujian pengaruh jumlah *sink node*

No.	Jumlah *sink node*	*Free Memory Heap (bytes)*
1	10	6148
2	20	6148
3	30	6148



Gambar 8. Grafik pengujian pengaruh jumlah *sink node*

Hasil analisis yang didapat dari pengujian jumlah *sink node* terhadap *memory heap* yang tersisa, dapat dijelaskan bahwa jumlah atau banyaknya *sink node* yang melakukan *request* pada proses *pull message* tidak berpengaruh terhadap *memory heap*. Sama seperti pengujian sebelumnya, hal ini disebabkan karena tidak ada mekanisme sinkronisasi dan penjadwalan *thread* menyebabkan proses pencatatan *memory heap* yang tersisa tidak dapat dilakukan pada waktu yang benar-benar bersamaan dengan proses penerimaan *request* data dari beberapa *sink node* sekaligus .

3.3. Pengujian pengaruh besar data terhadap *pull message time*

Tabel 3. Tabel hasil pengujian waktu *request* dan *response*

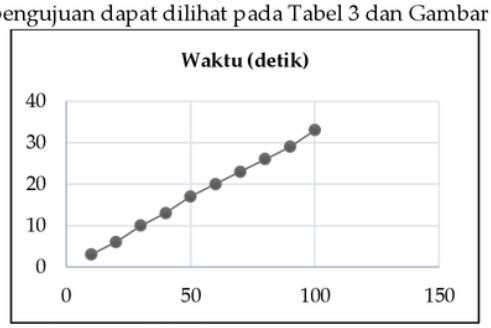
No.	Ukuran Data (Kilobyte)	Waktu (detik)
1.	10	3
2.	20	6
3.	30	10
4.	40	13
5.	50	17
6.	60	20
7.	70	23
8.	80	26
9.	90	29
10.	100	33

Pengujian ini dilakukan pada *sink node*. Tujuan pengujian ini dilakukan untuk mengetahui apakah besarnya data akan berpengaruh terhadap waktu *request* dan *response* pada proses *pull message*, yang dicatat oleh *sink node*. Skenario pada pengujian ini dilakukan melalui proses *request* (*pull* data sensor) yang dilakukan oleh *sink node* ke *sensor node*. Waktu yang dicatat adalah selisih waktu ketika *sink node* melakukan *request*, dan menerima *response* data dari *sensor node*. Dalam pengujian ini data yang

http://doi.org/10.26594/register.v3i2.699

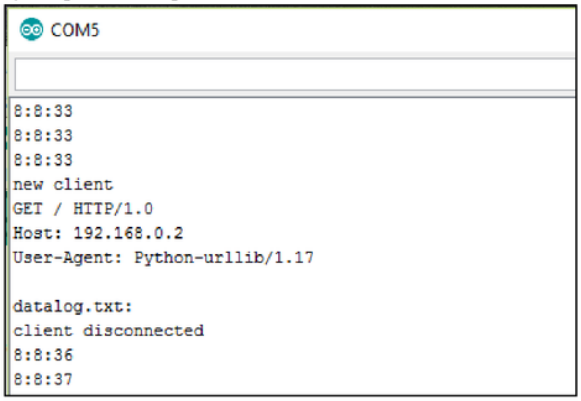
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

dikirimkan dibuat bervariasi, yaitu ukuran data 10 KB, 20 KB, 30 KB, 40 KB, 50 KB, 60 KB, 70 KB, 80 KB, 90 KB, dan 100 KB. Hasil pengujuan dapat dilihat pada Tabel 3 dan Gambar 9.



Gambar 9. Grafik hasil pengujian *request response time*

Hasil analisis yang didapat dari pengujian ini didapat bahwa terdapat jeda waktu yang berbeda antara proses pengiriman *request* data yang dilakukan oleh *sink node*, hingga menerima *response* data dari *sensor node*. Semakin besar ukuran data yang dipesan semakin besar jeda waktu proses *request* dan *response*. Dapat dilihat pada Gambar 9 bahwa kenaikan waktu proses *request* dan *response* terjadi secara linier sering dengan kenaikan jumlah data yang dipesan oleh *sink node* melalui proses *pull message*. Dari pengujian yang dilakukan juga didapatkan hasil bawah ketika proses penerimaan *request* dan pengiriman respon berjalan pada *sensor node* (Arduino), proses yang lain akan mengalami *interupt*. Proses terjadinya *interupt* dapat dilihat pada Gambar 10.



Gambar 10. Proses *interupt* yang terjadi pada *sensor node*

4. Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan pada penelitian ini maka didapatkan beberapa kesimpulan sebagai berikut:

1. Mekanisme *pull message* dengan menggunakan *Restful Web Service* dapat diimplementasikan pada proses komunikasi di dalam WSN. *Sensor node* (Arduino) dapat digunakan sebagai *web server* dan bisa menangani *request* dari banyak *client* melalui pemanfaatan *thread* sederhana, sedangkan *sink node* (Raspberry Pi) dapat melakukan proses *pull message* dengan menggunakan protokol *Restful Web Service* pada komunikasi WSN.
2. *Memory heap* yang tersisa digunakan untuk mengukur kinerja dari *sensor node* yang memanfaatkan metode *pull message* dengan protokol *restful*. Hasil pengujian didapat bahwa *memory heap* tidak terpengaruh dengan proses *sensing*, ukuran data, dan jumlah *sink node* dikarenakan tidak adanya mekanisme sinkronisasi dan penjadwalan *thread*. Sedangkan dalam pengujian waktu *request* dan *response*, ukuran data yang akan dikirim sangat berpengaruh terhadap waktu *request* dan *response* dari *sink node* ke *sensor node*.

https://doi.org/10.26594/register.vi.idpublikasi

ISSN 2502-3357 (*Online*)
R. A. Hidayatullah dkk./Register 3(2) 65-74 ISSN 2503-0477 (*Print*)

Hasil penelitian yang telah dilakukan masih jauh dari sempurna dan masih terdapat kekurangan. Diantaranya adalah masih tingginya waktu *request-response* antara *sink node* dengan *sensor node*. Untuk pengembangan penelitian berikutnya perlu untuk ditambahkan metode atau solusi yang dapat mengefisiensikan mekansime *pull message* (*request-response* data hasil *sensing*) sehingga waktu *request* dan *response* tidak terlalu berpengaruh siginifikan terhadap besarnya ukuran data hasil *sensing* yang dikirim.

5. Referensi

Barroca, N., M.Borges, L., J.Velez, F., Monteiro, F., Górski, M., & Castro-Gomes, J. (2013). Wireless sensor networks for temperature and humidity monitoring within concrete structures. *Construction and Building Materials, 40*(2013), 1156-1166.

Burgstahler, D., Lampe, U., Richerzhagen, N., & Steinmetz, R. (2013). Push vs. Pull: An Energy Perspective (Short Paper). *Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on* (pp. 190-193). Koloa: IEEE.

Chaudhary, D. D., Nayse, S. P., & Waghmare, L. M. (2011). Application of wireless sensor networks for greenhouse parameter control in precision agriculture. *International Journal of Wireless & Mobile Networks (IJWMN), 3*(1), 140-149.

Ferdoush, S., & Li, X. (2014). Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications. *Procedia Computer Science, 34*(2014), 103-110.

Georgitzikis, V., Akribopoulos, O., & Chatzigiannakis, I. (2012). Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks. *IEEE Latin America Transactions, 10*(3), 1686-1689.

Hamad, H., Saad, M., & Abed, R. (2010). Performance Evaluation of RESTful Web Services for Mobile Devices. *International Arab Journal of e-Technology, 1*(3), 72-78.

Käbisch, S., Peintner, D., Heuer, J., & Kosch, H. (2010). Efficient and Flexible XML-Based Data-Exchange in Microcontroller-Based Sensor Actor Networks. *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on* (pp. 508-513). Perth: IEEE.

Leguay, J., Lopez-Ramos, M., Jean-Marie, K., & Conan, V. (2008). An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks. *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on* (pp. 740-747). Montreal: IEEE.

Maarif, A. F. (2016). System Monitoring And Controlling Water Nutrition aquaponics Using Arduino Uno Based Web Server. *Kinetik, 1*(1), 39-46.

Othman, N. Y., Glitho, R. H., & Khendek, F. (2007). The Design and Implementation of a Web Service Framework for Individual Nodes in Sinkless Wireless Sensor Networks. *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on* (pp. 941-947). Las Vegas: IEEE.

Sawidin, S., Melo, O. E., & Marsela, T. (2015). Monitoring Kontrol Greenhouse untuk Budidaya Tanaman Bunga Krisan dengan LabView. *JNTETI (Jurnal Nasional Teknik Elektro dan Teknologi Informasi), 4*(4), 236-242.

6-Implementasi Pull Message dengan menggunakan Restful Web Service pada komunikasi Wireless Sensor

ORIGINALITY REPORT

19%	19%	10%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	doaj.org Internet Source	10%
2	Raden Budiarto. "Analisis faktor adopsi aplikasi mobile berdasarkan pengalaman, usia dan jenis kelamin menggunakan UTAUT2", Register: Jurnal Ilmiah Teknologi Sistem Informasi, 2017 Publication	5%
3	Submitted to Universitas Pendidikan Indonesia Student Paper	3%
4	www.scilit.net Internet Source	2%

Exclude quotes	Off	Exclude matches	< 2%
Exclude bibliography	Off		